Pergunta 2

Experiência 2.1

O programa recebe dois argumentos, o segredo e o número de entidades pelas quais o segredo será distribuído, N, e gera N chaves secretas com base no segredo fornecido.

Ao reconstruir o segredo são necessárias exatamente todas as componentes nas quais o segredo foi dividido. Caso sejam fornecidas menos ou mais componentes que o total de componentes em que o segredo foi dividido, o resultado de reconstruir o segredo não irá corresponder ao segredo original.

Experiência 2.2

O programa recebe três argumentos, o segredo, o número de partes necessárias para reconstruir o segredo, K, e o número de entidades pelas quais as chaves secretas serão distribuídas, N, onde 0 < K <= N. Por outras palavras, N é o número de chaves secretas que serão geradas.

Com isto o programa gera N chaves secretas, uma para cada entidade, sendo apenas necessárias K dessas chaves para reconstruir o segredo original. Caso sejam fornecidas menos do que K chaves secretas, não é possível reconstruir o segredo (erro por falta de segredos). Se forem fornecidas K ou mais chaves secretas, o programa de reconstrução apenas irá analisar as K primeiras chaves fornecidas descartando as restantes. Se alguma das K primeiras chaves for repetida, é retornado um erro a informar qual a linha em que se encontra a chave repetida. Se as K primeiras chaves fornecidas forem válidas e não repetidas, o segredo será gerado com sucesso. De notar que estas chaves podem ser dadas por qualquer ordem.

Pergunta P2.1

A)

Para dividir o segredo, recorremos ao programa *createSharedSecret-app.py*. Este programa recebe o número de partes em que o segredo vai ser dividido, *number_of_shares*, o número de partes necessárias ara reconstruir o segredo, *quorum*, um id para identificar unicamente cada segredo de modo a saber a que segredo pertencem as diferentes partes, *uid*, e um ficheiro "*.pem*" contendo a chave privada dado que cada parte do segredo é devolvida num objeto *JWT* assinado, em base 64. Como resultado, o programa vai gerar N componentes (chaves secretas).

Assim sendo, para dividir o segredo "Agora temos um segredo extremamente confidencial" em 8 partes, com *quorum* de 5 para reconstruir esse segredo, começamos por gerar o par de chaves (ficheiro "*.pem*") com o comando "*openssl genrsa -aes128 -out mykey.pem 1024*" utilizando a *passphrase* "123456".

```
root@CSI:~/Desktop/ES/TPs/guiao1/ShamirSharing# openssl genrsa -aes128 -out mykey.pem
1024
Generating RSA private key, 1024 bit long modulus
.......+++++
.....................+++++
e is 65537 (0x010001)
Enter pass phrase for mykey.pem:
Verifying - Enter pass phrase for mykey.pem:
```

De seguida executamos o programa descrito anteriormente, *createSharedSecret-app.py*, como podemos observar pelo seguinte comando: *python createSharedSecret-app.py 8 5 1 mykey.pem*

Output:

```
root@CSI:~/Desktop/ES/TPs/guiao1/ShamirSharing# python createSharedSecret-app.py 8 5 1 mykey.pem
Private key passphrase: 123456
Secret: Agora temos um segredo extremamente confidencial

Component: 1
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjEtN2M5OTEzNjAxN2VmMzdhNDVlYjU5MmI2NDFjZTQyNzE4ZDkyYzg0MmVhOGRiZWIwYzQzZWVlYjgwYWVkMGQ5Mzk4NDczODZlN2MyOTg1MjFh
MDA5YTQ2NjVmZjY1MjMwIiwgIjEiLCA1LCA4LCAiOWQ2NzY1MWI1ZWU5NGU1Y2EwOWM0NDNmMDUyMTBjNjc1ZDJjNDZmMTM2OGJkNGMyMjBlMWY2MDhlZWQ4OTNhNCJdfQ.Mp9L6E2AGxU2_nsie8wH
mho9uekpg71aQ-JOnEPkVmU-9n1LpjkL3Obz7sGObLytr4rIHMPKC3UI9416j1RtaTTN8ngpjvM2T1chK435qAIemqZSuass6-PHR3hPp2v3f2OiO8sQryDKYJ-sKxyx9SbXFV5gUW5KsBt9rRT7gVM
Component: 2
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjItMTRlODAwZWU5MzJkNzFhYjYyOWE0Y2Q0MjZkZGNiMTVhOGZiYmZiOWQxNjM1ZTU0NzgwMjIwZjNjYjA0ZTZlNWU0NDc4NDY1YmU1MGU0MTc0
ZDI1NzUzMWQxZGI0MTE3IiwgIjEiLCA1LCA4LCAiYTA3MDViMzRhNzJmMDJiN2Q3NGU5YmM1YTEyYTI4NzI5Nzk2MTQyYzNhNWVlOTI4MWViYWM5MjUyZDVjNDdhOCJdfQ.a-tG71-BGwYUv_RRUyOk
q8T_On8BIl-utpim2Kt1rbddZRuAzoch7LonExHmtR-OofwIK4NdvB06jxwRn3w0cm1i6cUDuAEGgBRxsyK16qfUFgnWCXs8XhgEm1TmnHJhqmYv_KnBtrjrVdY3lNmTWWr0pIKKB91VEqHVwD_hEkk
Component: 3
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjMtZjY1NDY2ZTA1OTk1ODlhZjdiZjEyNTk3NDFkNTlhNmRjZjBkMGQzYjhjNDIzM2RlMjU0ZDc5Y2JkNzlhYmRlZTk0ZDU2ZDBlN2UyZmEyODdh
NjU2YTYzNTM1YzUwNDQ3IiwgIjEiLCA1LCA4LCAiNDQ2ZTI1NzVkNmViYTdiNTI2MzM2YTc3N2JkY2MwYzg3ZDQ2MDIxMWIxNzU3MzRlODdiYTVlMjBmNzQ0ZjgwYSJdfQ.E5rwfejCj0y8QS5w93MS
M88-qnkiA6s94Bm2sPVJPd5wIpQRGGULnpXJ1Rn222lGy0Uv0kZEKqFL6R8WbSPtOfzA2w4_kgCYcqVCLU7PpV9BvePPTHl7p9ItGo-b5o1APF5YHlwv55CpFED3mvLXSVq6UL0h4B8J6Df6irLumY4
Component: 4
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjQtZGUwNTkwMmQ5MTQ2NjlmN2U1ZGM0ZjUzZjg0NDliOTc4YjE3ZmIzMWM2MTk3YTZlYmNlNDRlYTZmZjE0ZDMzNjdmY2I4MGYwYzhkYjM2YjRj
MjNjZThlZjgwZGQ1NzcxIiwgIjEiLCA1LCA4LCAiM2FiYzU0YTIzODQ2ZmRiMDBlMDliZTdlYzRkOGYzYzgyYmZlZGMxYjQ4ZjFjZTU5MjU4Mzc3M2MxYTI5ZmQ0MSJdfQ.JjXfOPfHEBsiSG3y_K38
oFo4TlX8UhhCOTMKa1b7rmRIANQfFbXALzqjrCGO6jsVaYyDl-n5c0gDjhumMlJeNTdszsV6Ww_6q0bkF3Nrwez2KEZdi4rGG03Bt8G2Vmb8OyKlOLAS92bNjmiPP3jKTB61N09SPfCwsTRgDjjyIZI
Component: 5
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjUtNWE0OWU0MDNmZmMzN2ZkZTEzY2NhYTk1ZjgwZWM0YmZkYzU0NjlmZTYxNzU3M2JhOGViZGQ5OWZiNDRkZTMzZDBlN2ViOTE0NjQzNDc1ZTI5
ODc4ZDFmYTIwY2FlMjIyIiwgIjEiLCA1LCA4LCAiNWY4Y2E2ZjUwN2U3YjA1ZDJiNTViZjg2OTMxZDc3ZjZmNDE2OTU5NjBhNzBjYjJiMjJkOTc2YzM2YmE5NmMxOCJdfQ.PYZ1MHjrr4lb5sJHN8EO
AB_C6KSOkuiFlBhxf1y0GSxTYbf6RkqLSnKXis-ZH7_GniuuBxI9PEXifXi4cmtwaJuM5-YGrrItkVy_RrYvfLLjHmgSF14bpLdaFYec2q2C29xJiaHV_8MOjwHXFAoIondHPC2ZEHqUFblJQ8A3tRg
Component: 6
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjYtY2E5NmU1YzcwOWY0YmJjZGIxODFjNjIwMzc2ZDE3MjMzNWUwY2ZmYjc4N2Y2ODA5NGEwNTM5ODQwZTk1MjY3ODNkYmU3OTAwY2VlNzk0NTcw
MWFiZDkwOGZiYjEzNWY1IiwgIjEiLCA1LCA4LCAiMTg4OTVkMWNlMmFjZjBlNDIwZmNhNTExNWQwYmNkZDczOTM4M2IwOTJhYjk5M2E5NGRlMmFkNjgzYmJmMGFhNSJdfQ.PQvcTis0JNd16yMSw2P6
YcCPf0WUCig6pgbKxmSQWYRY428qdoE34BEykw-CrqIAuuXxe5MlIbOtMYouL6-UPJhxNVW02YxmOASgGMCR4Af3jKiipOqGQ3X5x9xNy5ts6A_oh7dzr-y7YAOrbzXUyH3z6Aupi8yFJPs1OizMxYY
Component: 7
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjctNWY4OTM1MGZiNDI2OTE0NGEzMDlkZWVjZjRlZDlmMGQ3ZWMyMzk3YjVjZmRhNjM1ZmQxOTcxZDVjOWFlNTE1NDlkZDQ4ZWJkNDA2YzI1NTli
Nzc4NThlYTcwMzBjYWY3IiwgIjEiLCA1LCA4LCAiMWNmNTE0MTA0OTExYzZkMDk1NWYyOWM1MTgxZWJkZjlmOGExZGUzODhlZjE5NWYwM2FjYjMxMzk0YTVjMTc1NCJdfQ.hSYKu1Upl5GGCaePo1WI
N53E_W1f3-S0urDrlZ-XDCbDDu5b3sqSr0cs2JgH9jMNwoSuLk3zN4CUg8naydUIvi2LvW-FAizIvjOodhZsUm34pXz_nHiu3LocTj_6EDc7ZCtX30xeSt9AgV8MqtbnI0VZmhR92-beNkBFg-LJehk
Component: 8
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjgtMWFlNDhkYWNhMjA5ZjZkMzA0YzFlMDJkYjc3MzczZGExMWU0NDhjODk4NTM4M2FmMTU4ZDZhY2E0NWQxOTQzNTUyODUzMmQwYTkwNTFiMzQ1
NDgxOGY4NzU1NjcwNTExIiwgIjEiLCA1LCA4LCAiNjI4NmUyNTUzNzA0NDJkMjc1YzRkNjIwMDlmMTY1ZmVkNWE2ZDQ3NDNlMTJiMzFmNDg2M2ZlYzA0YWRkZTNmNCJdfQ.JR-RIK9w3W_7QTJZBX7p
Jwl5C_scPXEIEHzDS62GFZJAIVrVY7EZXyjY_4HrIJcwTvjh7yoZIgwGJqccUMy5KjoMwOv6wUYxWixcKW4MY-UnH6WUt-EbW4obXERB33rfNCXmUtOjIYLr2td7WFpDijbJyPrsuEaU8x-EN6oXXeg
```

Onde a *passphrase* é a que foi fornecida na geração do par de chaves e o *secret* é o segredo original usado para originar as 8 partes (chaves privadas) a serem distribuídas pelas 8 entidades. Na imagem apresentada acima podemos ver as 8 partes (chaves privadas) originadas a partir do segredo.

Dado que para recuperar o segredo é necessário fornecer um certificado, usamos o comando "*openssl req -key mykey.pem -new -x509 -days 365 -out mykey.crt*" para gerar o certificado correspondente ao par de chaves gerado anteriormente.

Além do certificado, tanto o programa *recoverSecretFromComponents-app.py* como o programa *recoverSecretFromAllComponents-app.py* precisam de receber como argumentos o número de partes que serão usadas para recuperar o segredo inicial e o id do segredo (argumento *uid* fornecido ao programa *createSharedSecret-app.py*). Caso não sejam apresentadas o número mínimo de partes necessárias para recuperar o segredo original, é retornado um erro a informar que o número de partes fornecidas é inferior ao *quorum*. Caso alguma das partes referidas seja repetida, é retornado um erro a informar que foram fornecidas componentes repetidas. Por último, caso alguma das partes fornecidas não apresente um formato válido ou não tenha sido assinada com a chave privada associada à chave pública do certificado fornecido, é retornado um erro a indicar que pelo menos um destes dois casos se verificou. Em todos os casos descritos acima, não é possível recuperar o segredo. Se forem fornecidas pelo menos *quorum* partes válidas, todas distintas entre si e que tenham sido assinadas com a chave privada associada à chave pública do certificado, é possível recuperar o segredo original.

B)

Enquanto que no programa *recoverSecretFromComponents-app.py* é necessário fornecer apenas um número de partes igual ou superior ao *quorum* indicado ao gerar estas partes, no programa *recoverSecretFromAllComponents-app.py* é necessário fornecer todas as partes geradas.

Por outras palavras, se o segredo for dividido em 8 partes e o *quorum* for de 5 partes, caso sejam fornecidas:

- menos de 5 partes válidas e distintas entre si, ambos os programas irão retornar um erro por falta de partes;
- entre 5 e 7 partes válidas e distintas entre si, o primeiro programa irá retornar o segredo inicial enquanto que o segundo programa irá retornar um erro por falta de partes;
- as 8 partes, e estas sejam válidas e distintas entre si, ambos os programas irão retornar o segredo inicial.

Nota 1: Entenda-se por parte válida uma parte que apresente o formato adequado e que tenha sido assinada com a chave privada associada à chave pública do certificado fornecido.

Nota 2: Caso sejam fornecidas mais de 8 partes, pelo menos uma delas é obrigatoriamente inválida ou repetida.

O primeiro programa, *recoverSecretFromComponents-app.py*, pode ser usado em situações mais banais cujas consequências não sejam muito drásticas. Já o segundo programa, *recoverSecretFromAllComponents-app.py*, pode ser necessária em situações de extrema importância cuja operação desencadeada pela utilização segredo completo tenha consequências drásticas. Por exemplo, a utilização de lança-mísseis pode ser classificada tendo em conta o tipo de mísseis lançados. Para operar com mísseis menos perigosos pode ser necessária a participação de 5 dos 8 detentores das chaves. Para operar com mísseis mais perigosos deve ser necessária a participação de possivelmente todos os 8 detentores das chaves. Outro exemplo trata-se de operações de transferência (mais concretamente remoção) de dinheiro de um depósito de uma empresa, onde quantias normais necessitem de apenas uma porção das chaves enquanto que quantias mais elevadas necessitem das chaves (aprovação) de todos os responsáveis da empresa.